MDS431\_LAB1(Part 2-and Part-3)\_2448040.R

**Course code: MDS 431 – Time Series and Forecasting Techniques**

**Exercise Lab: 1**

**Date: 26/06/25**

**QUESTION:**

**Using ACF Plots, comment on the data you have selected in your previous practical. Check for the stationarity of the series. If it is not stationary, convert it into a stationary series.**

**ANSWER:**

**INTRODUCTION:**

Time series analysis is a method used to analyze data collected over time to identify patterns, trends, seasonality, and other structural elements. It is especially valuable in climatology, economics, and environmental sciences. This lab focuses on analyzing monthly mean temperature data in India to understand its variability and implications.

The dataset used in this analysis has been sourced from the **Indian Meteorological Department (IMD)** and made publicly available through the **Government of India Open Data Platform**:  
 [https://www.data.gov.in/resource/monthly-seasonal-and-annual-mean-temp-series-1901-2017](%20https:/www.data.gov.in/resource/monthly-seasonal-and-annual-mean-temp-series-1901-2017)

This dataset contains monthly, seasonal, and annual mean temperature series for all 12 months (January to December), over a long historical span from 1901 to 2017.

**DATASET DESCRIPTION:**

The dataset contains 117 annual observations (from 1901 to 2017), each with monthly mean temperature values across the year. The structure is as follows:

**YEAR**: The observation year (1901–2017)

**JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC**: Monthly mean temperature values (in degrees Celsius)

**ANNUAL**: The annual mean temperature

**JAN-FEB, MAR-MAY, JUN-SEP, OCT-DEC**: Seasonal mean temperatures

There are no missing values in the dataset, making it suitable for direct time series analysis.

**OBJECTIVE:**

The main objectives of the lab are as follows:

* To understand the structure and purpose of time series data.
* To load and preprocess a real-world dataset using R.
* To convert monthly temperature data into a proper time series object.
* To identify and interpret key time series components: trend, seasonality, and irregular variation.
* To determine whether the time series follows an additive or multiplicative model structure.
* To assess the stationarity of the time series and apply appropriate transformations if necessary.

**METHODOLOGY/CODES:**

#Loading required libraries  
library(tseries)

## Warning: package 'tseries' was built under R version 4.4.3

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

library(forecast)

## Warning: package 'forecast' was built under R version 4.4.3

#Reading the dataset  
data <- read.csv("C:/Users/Neelanjan Dutta/OneDrive/Desktop/Time Series Forecasting/dataset.csv")  
  
#View first few rows  
head(data)

## YEAR JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC  
## 1 1901 17.99 19.43 23.49 26.41 28.28 28.60 27.49 26.98 26.26 25.08 21.73 18.95  
## 2 1902 19.00 20.39 24.10 26.54 28.68 28.44 27.29 27.05 25.95 24.37 21.33 18.78  
## 3 1903 18.32 19.79 22.46 26.03 27.93 28.41 28.04 26.63 26.34 24.57 20.96 18.29  
## 4 1904 17.77 19.39 22.95 26.73 27.83 27.85 26.84 26.73 25.84 24.36 21.07 18.84  
## 5 1905 17.40 17.79 21.78 24.84 28.32 28.69 27.67 27.47 26.29 26.16 22.07 18.71  
## 6 1906 17.50 19.14 22.21 26.53 29.06 28.02 27.46 26.82 26.23 24.75 21.93 19.55  
## ANNUAL JAN.FEB MAR.MAY JUN.SEP OCT.DEC  
## 1 24.23 18.71 26.06 27.30 21.92  
## 2 24.33 19.70 26.44 27.18 21.49  
## 3 23.80 19.05 25.47 27.17 21.27  
## 4 23.86 18.66 25.84 26.83 21.42  
## 5 23.71 17.58 24.99 27.37 21.48  
## 6 24.12 18.37 25.93 27.15 22.08

#Checking the structure and column names  
str(data)

## 'data.frame': 117 obs. of 18 variables:  
## $ YEAR : int 1901 1902 1903 1904 1905 1906 1907 1908 1909 1910 ...  
## $ JAN : num 18 19 18.3 17.8 17.4 ...  
## $ FEB : num 19.4 20.4 19.8 19.4 17.8 ...  
## $ MAR : num 23.5 24.1 22.5 22.9 21.8 ...  
## $ APR : num 26.4 26.5 26 26.7 24.8 ...  
## $ MAY : num 28.3 28.7 27.9 27.8 28.3 ...  
## $ JUN : num 28.6 28.4 28.4 27.9 28.7 ...  
## $ JUL : num 27.5 27.3 28 26.8 27.7 ...  
## $ AUG : num 27 27.1 26.6 26.7 27.5 ...  
## $ SEP : num 26.3 25.9 26.3 25.8 26.3 ...  
## $ OCT : num 25.1 24.4 24.6 24.4 26.2 ...  
## $ NOV : num 21.7 21.3 21 21.1 22.1 ...  
## $ DEC : num 18.9 18.8 18.3 18.8 18.7 ...  
## $ ANNUAL : num 24.2 24.3 23.8 23.9 23.7 ...  
## $ JAN.FEB: num 18.7 19.7 19.1 18.7 17.6 ...  
## $ MAR.MAY: num 26.1 26.4 25.5 25.8 25 ...  
## $ JUN.SEP: num 27.3 27.2 27.2 26.8 27.4 ...  
## $ OCT.DEC: num 21.9 21.5 21.3 21.4 21.5 ...

names(data)

## [1] "YEAR" "JAN" "FEB" "MAR" "APR" "MAY" "JUN"   
## [8] "JUL" "AUG" "SEP" "OCT" "NOV" "DEC" "ANNUAL"   
## [15] "JAN.FEB" "MAR.MAY" "JUN.SEP" "OCT.DEC"

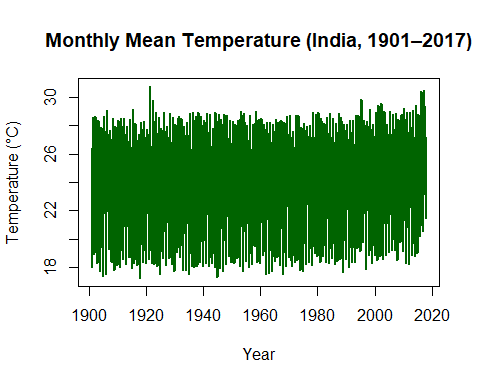
#Check for missing values  
sum(is.na(data))

## [1] 0

#Convert wide format (YEAR, JAN–DEC) to a long vector  
monthly\_temps <- as.vector(t(data[, 2:13])) # Take JAN–DEC columns row-wise  
start\_year <- data$YEAR[1] # Starting year, should be 1901  
  
#Create monthly time series object  
temp\_ts <- ts(monthly\_temps, start = c(start\_year, 1), frequency = 12)  
  
#Confirm it's a time series object  
class(temp\_ts)

## [1] "ts"

#Plot the time series  
ts.plot(temp\_ts,  
 main = "Monthly Mean Temperature (India, 1901–2017)",  
 ylab = "Temperature (°C)",  
 xlab = "Year",  
 col = "darkgreen", lwd = 2)



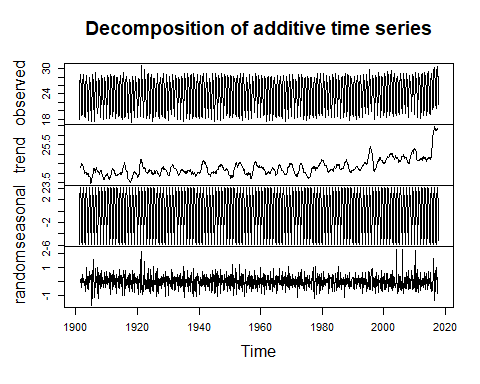
**INTERPRETATION:**

The plot displays the monthly mean temperature (in centigrade) for India from 1901 to 2017. The graph shows three major components of a time series. They are:

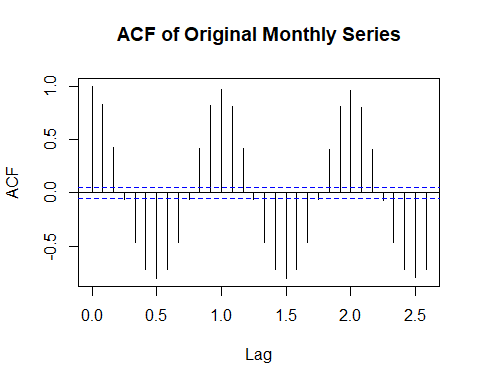
1. T**rend:** There is a subtle upward shift in the upper range of temperatures over the decades, especially after 2000, indicating a gradual long-term warming trend.
2. **Seasonality:** The pronounced vertical spread within each year reflects regular seasonal fluctuations, with higher temperatures in some months and lower in others, repeating consistently every year.
3. **Random variation:** The irregular spikes and dips scattered throughout the series represent random fluctuations and anomalies not explained by trend or seasonality, likely due to short-term weather events or measurement noise.

We decompose the plot to see the components clearly.

#Decompose time series into trend, seasonality and random component  
decomp <- decompose(temp\_ts, type = "additive")  
plot(decomp)



#ACF plot to check for autocorrelation  
acf(temp\_ts, main="ACF of Original Monthly Series")



**INTERPRETATION**

The ACF plot clearly indicates non-stationarity due to several key characteristics in the lag values. The autocorrelation at lag 1 is extremely high (approximately 0.8-0.9), which is a classic indicator of non-stationary data. Additionally, the ACF values decline very slowly across multiple lags rather than dropping to zero quickly, and many values exceed the confidence intervals (blue dashed lines) [-0.2, 0.2], confirming that the series lacks stationarity.

**Test to show that the result is not stationary:**  
There are various tests to check whether a time series is stationary or not. One such test is the ADF test (Augmented Dickey-Fuller Test)

Let us set up the null hypothesis

H0: The time series is non-stationary

Against the alternative hypothesis

H1: The time series is stationary

#ADF test for stationarity  
adf\_result <- adf.test(temp\_ts)  
  
if (adf\_result$p.value < 0.05) {  
 print("ADF Test: Time series is stationary.")  
} else {  
 print("ADF Test: Time series is NOT stationary.")  
}

## [1] "ADF Test: Time series is NOT stationary."

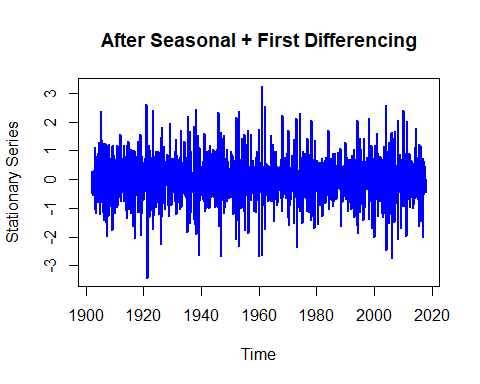
Since the time series is not stationary, we need to make it stationary. To make it stationary, we need to remove the components (seasonal and trend). The seasonal component can be removed by the **method of seasonal differencing**.

**Seasonal differencing:** It means removing repeating yearly patterns from your data, like subtracting January this year from January last year, and doing the same for every month.

#Remove seasonality via seasonal differencing (lag = 12)  
temp\_seasonal\_diff <- diff(temp\_ts, lag = 12)

After seasonality has been removed, we are left with the trend component. To make the time series stationary, we need to remove this trend component also. There are three methods to remove the trend component. They are:

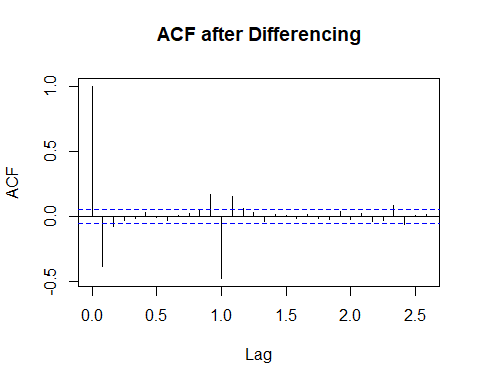
1. **Method of differencing:** It is used to eliminate the trend component from the time series.
2. **Method of least squares:** To estimate the parameters.
3. **Method of moving averages:** This method is used for smoothing out the trend component.  
     
   #Remove trend via regular differencing  
   temp\_stationary <- diff(temp\_seasonal\_diff)  
     
   #Plot transformed series  
   ts.plot(temp\_stationary,  
    main = "After Seasonal + First Differencing",  
    ylab = "Stationary Series",  
    col = "blue", lwd = 2)



**INTERPRETATION:**

The plot shows that the time series has undergone both seasonal differencing for removing the seasonal component and first differencing for removing the trend component, thus leaving behind a stationary time series. The fluctuations around a constant mean with no visible trend or periodic pattern indicate that only random variation (white noise) remains in the data.

#ACF plot of differenced series  
acf(temp\_stationary, main="ACF after Differencing")



**INTERPRETATION:**

The ACF plot after differencing shows that most autocorrelation values lie within the blue confidence bounds, indicating that the time series has little to no significant autocorrelation remaining. This suggests that the differencing process has effectively removed the trend and seasonality from the data, making it stationary. The absence of strong spikes at higher lags confirms that the series now primarily consists of random noise and is suitable for further modeling using a stationary-based technique.

#ADF test after differencing  
adf\_diff\_result <- adf.test(temp\_stationary)

## Warning in adf.test(temp\_stationary): p-value smaller than printed p-value

if (adf\_diff\_result$p.value < 0.05) {  
 print("ADF Test after differencing: Time series is now stationary.")  
} else {  
 print("ADF Test after differencing: Time series is still NOT stationary.")  
}

## [1] "ADF Test after differencing: Time series is now stationary."

# Summary statistics  
summary(temp\_ts)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 17.25 21.07 25.57 24.29 27.24 30.78

**INTERPRETATION:**

* **Minimum Temperature**: **17.25°C**, the lowest recorded value in the series, indicating a particularly cool observation period.
* **Maximum Temperature**: **30.78°C**, the highest recorded value, representing a significantly warm time.
* **Mean Temperature**: **24.29°C**, the average value across all observations, reflecting the overall central tendency of the temperature series.
* **Median Temperature**: **25.57°C**, slightly higher than the mean, indicating that the data may be mildly left-skewed with a few lower temperature values pulling the mean down.
* **1st Quartile (Q1)**: **21.07°C**, meaning 25% of the data points are below this value, representing relatively cooler observations.
* **3rd Quartile (Q3)**: **27.24°C**, indicating that 75% of the data falls below this temperature, and only 25% of observations are exceptionally warm.

**CONCLUSION:**

In this lab, a time series analysis was performed on India's monthly, seasonal, and annual mean monthly temperature data (January–December) from 1901 to 2017. The dataset was complete, with no missing values, and was successfully converted into a time series object in R. Initial analysis indicated non-stationarity, as observed through trends and autocorrelation patterns. To address this, both seasonal and first differencing were applied. After these transformations, the time series became stationary, as confirmed by the plot and ACF analysis showing only random fluctuations. Summary statistics showed a fairly symmetric distribution with moderate variability. Overall, the data was effectively pre-processed for time series modeling, and an additive model structure was considered suitable based on the consistent variance and absence of multiplicative effects.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***